Faster solver for multiple linear systems via Block Conjugate Gradient

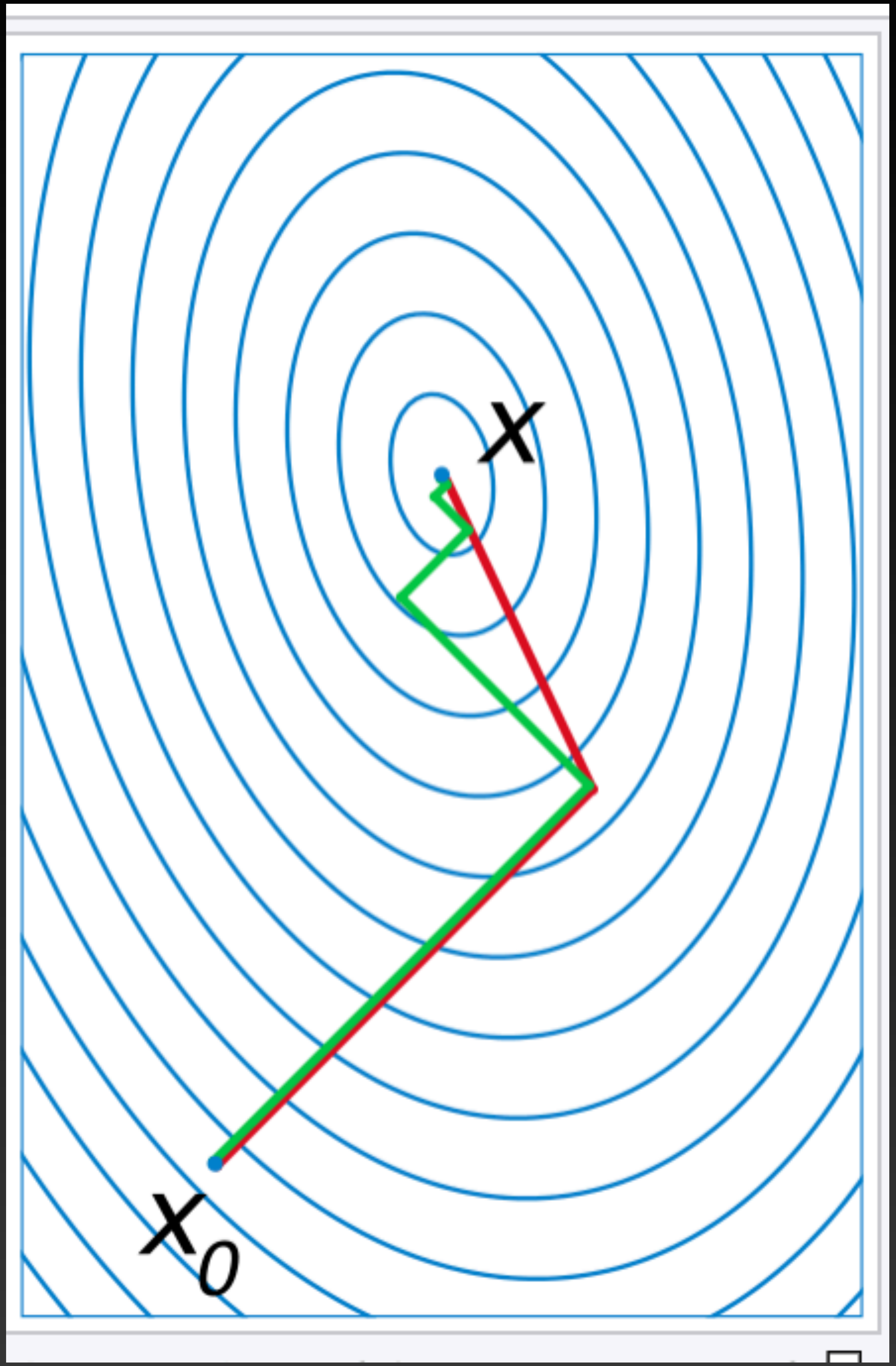
NYU's Courant Institute of Mathematical Sciences

William Lu Mentor: Florian Wechsung July-28th

Question

Find x such that Ax= b

- LU Decomposition: high computational cost
- Steepest Descent
- Conjugate Gradient: search directions are orthogonal



Idea: Block Conjugate Gradient

Sometimes we want to solve many problems at the same time.
i.e $AX = B, X \in R^{n \times l}, B \in R^{n \times l}$

Can we do better than just solving each of them separately? For instance, solving ℓ linear systems using Block Conjugate Gradient once instead of using CG ℓ times.

We have this hope because of the concept of memory communication cost and information sharing.

Less communication between CPU and memory; Share information between linear systems due to larger Krylov subspace.

CG & Block CG

Algorithm 1 CG

Input: Matrix A, a guessed solution x_0, a RHS b, and a threshold.
$r_0 = b - Ax_0$
if r_0 is smaller than the threshold, return x_0.
$p_0 = r_0$
while true **do**
  $\alpha_k = \frac{r_k^\top r_k}{p_k^\top A p_k}$
  $x_{k+1} = x_k + \alpha p_k$
  $r_{k+1} = r_k - \alpha A p_k$
  if r_{k+1} is smaller than the threshold **then**
    exit the loop
  else
    $\beta_k = \frac{r_{k+1}^\top r_{k+1}}{r_k^\top r_k}$
    $p_{k+1} = r_{k+1} + \beta_k p_k$
End Repeat
return x_{k+1}

Algorithm 2 Block CG

Input: Matrix A, a guessed solution X_0, a RHS B, and a threshold.
$R_0 = B - AX_0$
if R_0 is smaller than the threshold, return X_0.
$P_0 = R_0$
while true **do**
  $\Lambda_k = (P_k^\top A P_k)^{-1} R_k^\top R_k$
  $X_{k+1} = X_k + P_k \Lambda_k$
  $R_{k+1} = R_k - AP_k \Lambda_k$
  if R_{k+1} is smaller than the threshold **then**
    exit the loop
  else
    $\Phi_k = (R_k^\top R_k)^{-1} R_{k+1}^\top R_{k+1}$
    $P_{k+1} = R_{k+1} + P_k \Phi_k$
End Repeat
return X_{k+1}

- $Ax = b, AX = B$
- b, B : the RHS
- A : the positive def symmetric matrix
- r_k, R_k : k-th residual vectors
- p_k, P_k : k-th search direction
- x_k, X_k : k-th iteration
- α_k, Λ_k : k-th coefficient for step magnitude
- β_k, ϕ_k : k-th coefficient for search direction

- CG's Convergence Estimate:
- $e_k = x_k - x$
- $\kappa = \frac{\lambda_n}{\lambda_1}$, condition number
- Convergence theorem:
- $||e_k||_A \leq 2(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1})^k ||e_0||_A$
- $||e_k||_A = (e_k^T A e_k)^{\frac{1}{2}}$

- BCG's Convergence Estimate:
- $E_k = X_k - X$
- $\kappa_\ell = \frac{\lambda_n}{\lambda_\ell}$, where λ_ℓ is the ℓth largest eigenvalue of A
- Convergence theorem:
- $||E_k||_A \leq 2(\frac{\sqrt{\kappa_\ell} - 1}{\sqrt{\kappa_\ell} + 1})^k ||E_0||_A$
- $||E_k||_A = (E_k^T A E_k)^{\frac{1}{2}}$

Preconditioned CG & Preconditioned Block CG

Why preconditioning? What's the preconditioner?

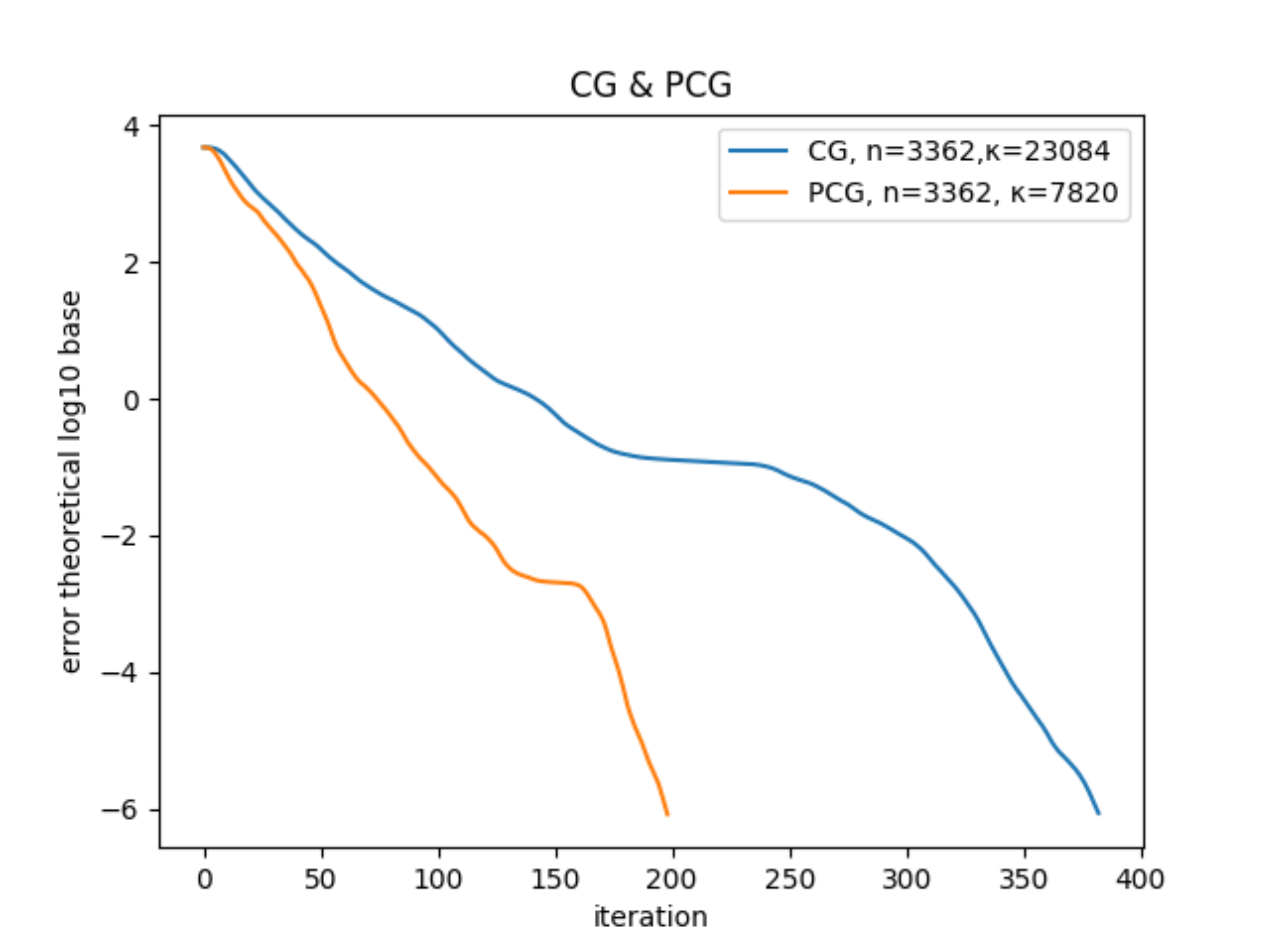
Algorithm 3 PCG

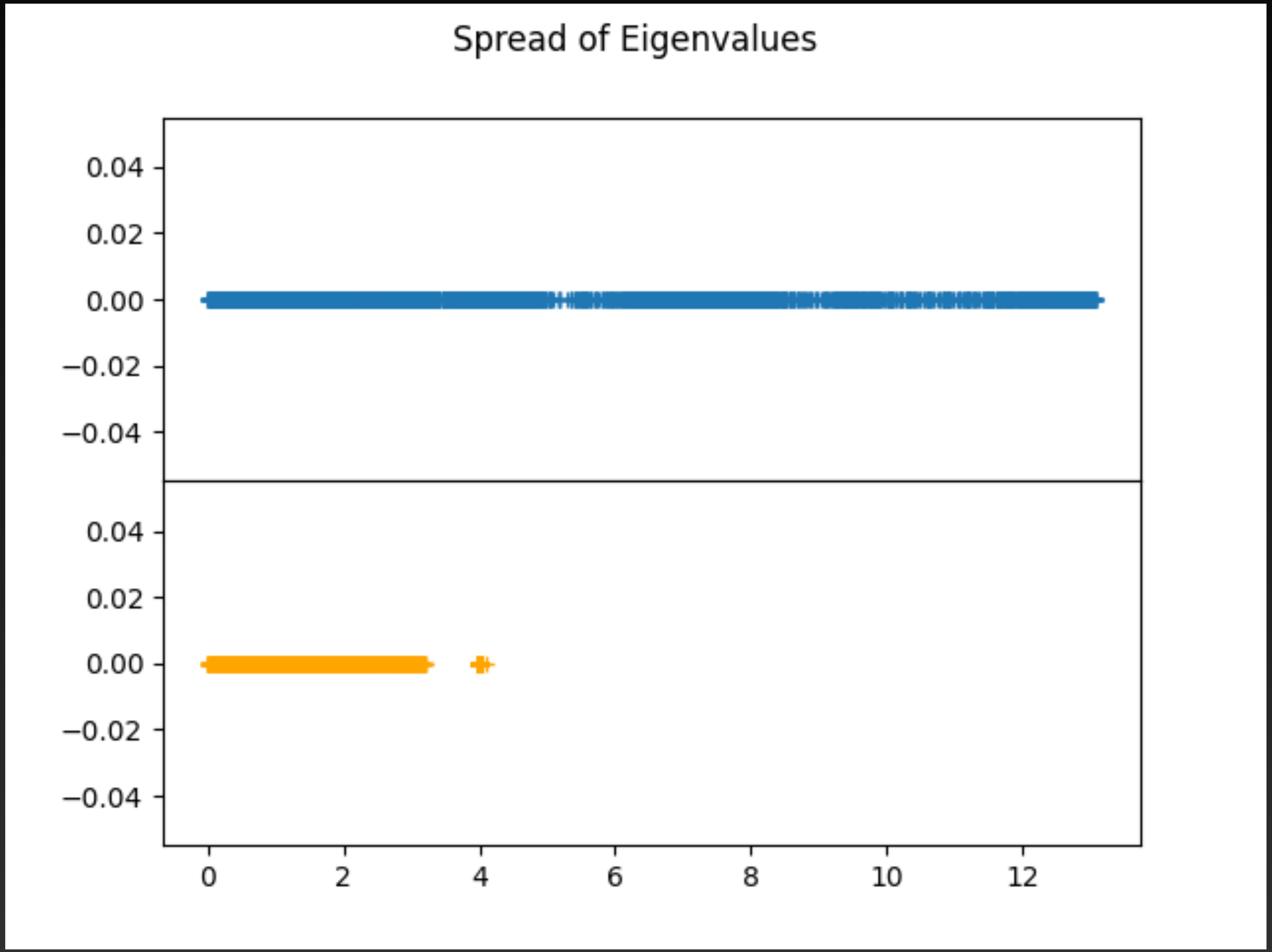
Input: Matrix A, a preconditioner M, a guessed solution x_0, a RHS b, and a threshold.
$r_0 = b - Ax_0$
$z_0 = M^{-1} r_0$
if r_0 is smaller than the threshold, return x_0.
$p_0 = r_0$
while true **do**
  $\alpha_k = \frac{r_k^\intercal z_k}{p_k^\intercal A p_k}$
  $x_{k+1} = x_k + \alpha p_k$
  $r_{k+1} = r_k - \alpha A p_k$
  if r_{k+1} is smaller than the threshold **then**
    exit the loop
  else
    $z_{k+1} = M^{-1} r_{k+1}$
    $\beta_k = \frac{r_{k+1}^\intercal z_{k+1}}{r_k^\intercal z_k}$
    $p_{k+1} = z_{k+1} + \beta_k p_k$
End Repeat
return x_{k+1}

Algorithm 4 PBCG

Input: Matrix A, a preconditioner M, a guessed solution X_0, a RHS B, and a threshold.
$R_0 = B - AX_0$
$Z_0 = M^{-1} R_0$
if R_0 is smaller than the threshold, return X_0.
$P_0 = R_0$
while true **do**
  $\Lambda_k = (P_k^\intercal A P_k)^{-1} R_k^\intercal Z_k$
  $X_{k+1} = X_k + P_k \Lambda_k$
  $R_{k+1} = R_k - A P_k \Lambda_k$
  if R_{k+1} is smaller than the threshold **then**
    exit the loop
  else
    $Z_{k+1} = M^{-1} R_{k+1}$
    $\Phi_k = (R_k^\intercal Z_k)^{-1} R_{k+1}^\intercal Z_{k+1}$
    $P_{k+1} = Z_{k+1} + \Phi_k P_k$
End Repeat
return x_{k+1}

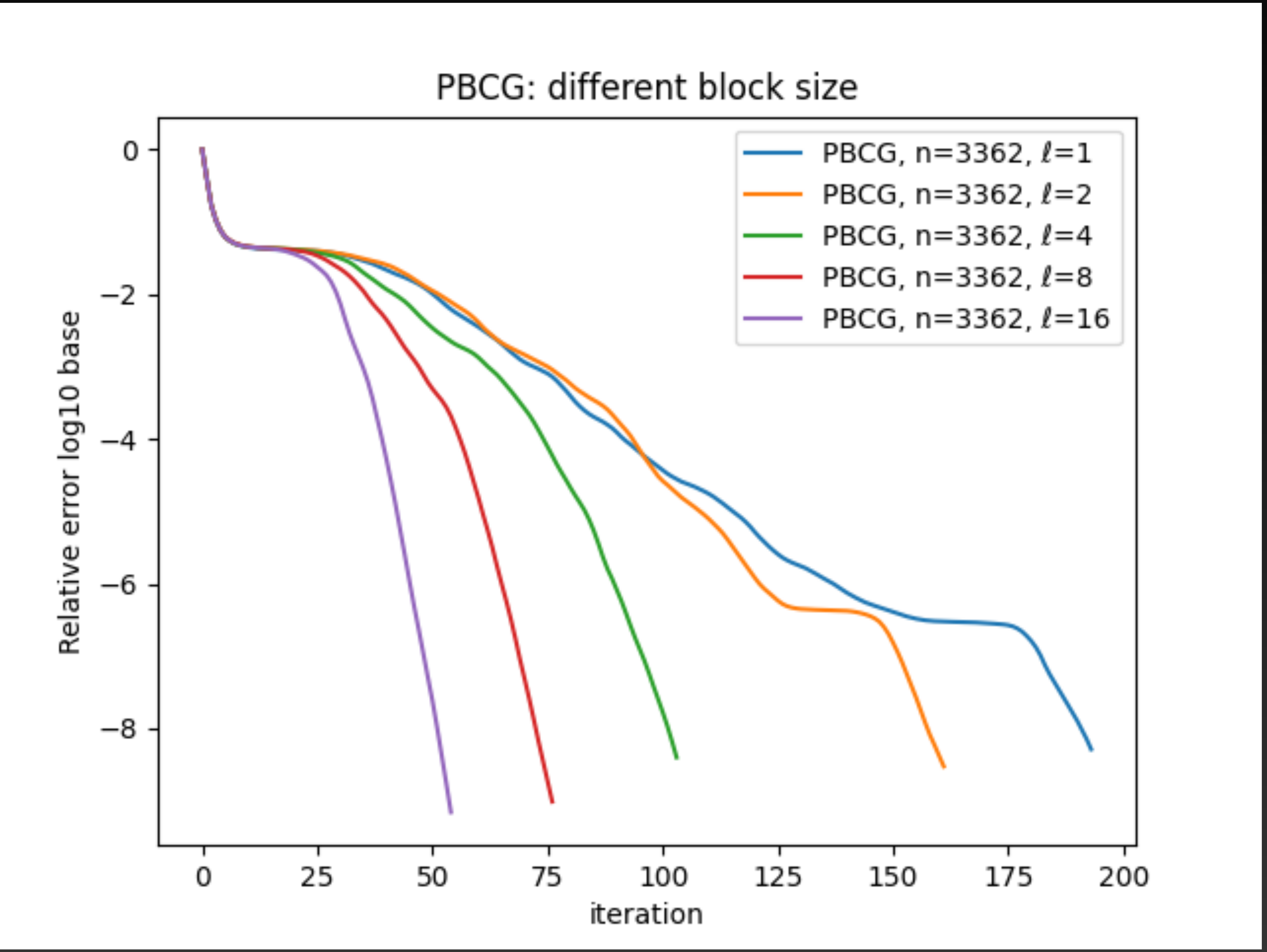
- $AX = B$
- M : preconditioner
- $M^{-1} \approx A^{-1}. M^{-1}A \approx I.$
- $M^{-1}AX = M^{-1}B$


CG & PCG
CG, n=3362,κ=23084
PCG, n=3362, κ=7820
error theoretical log10 base
iteration


Spread of Eigenvalues

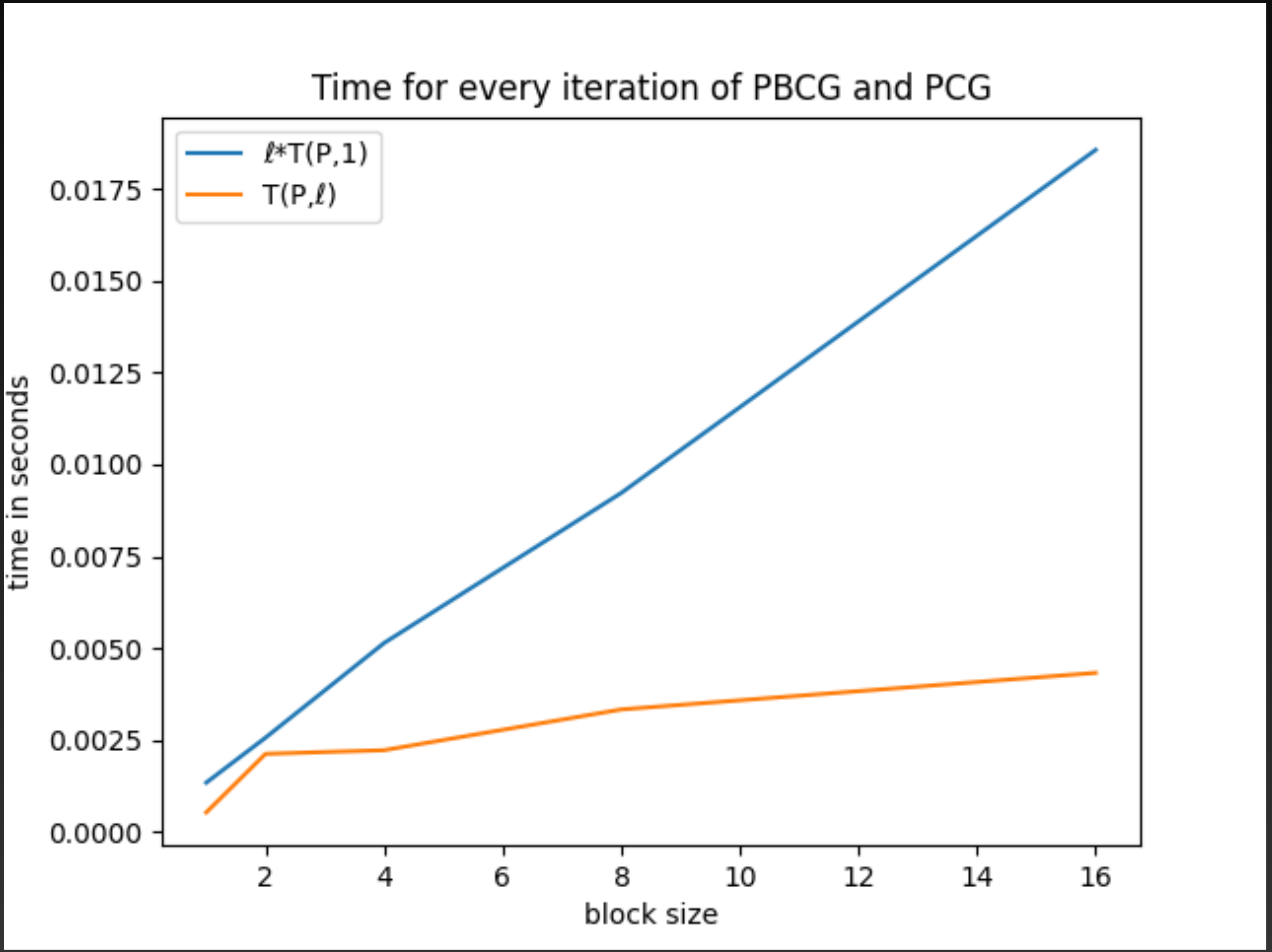
Number of iterations for Block is fewer

- matrix of size 3362
- ℓ is the block size
- $T_{\{PCG,\ell\}} = \boxed{Iter_{\{PCG,1\}}} \cdot T_{\{MatVec,1\}} \cdot \ell$
- $T_{\{PBCG,\ell\}} = \boxed{Iter_{\{PBCG,\ell\}}} \cdot T_{\{MatMat,\ell\}}$



$$A_{\{n\}}, n \in \{882, 3362, \mathbf{13122}\}$$

Fewer iterations. Each iteration is cheaper for large block size.

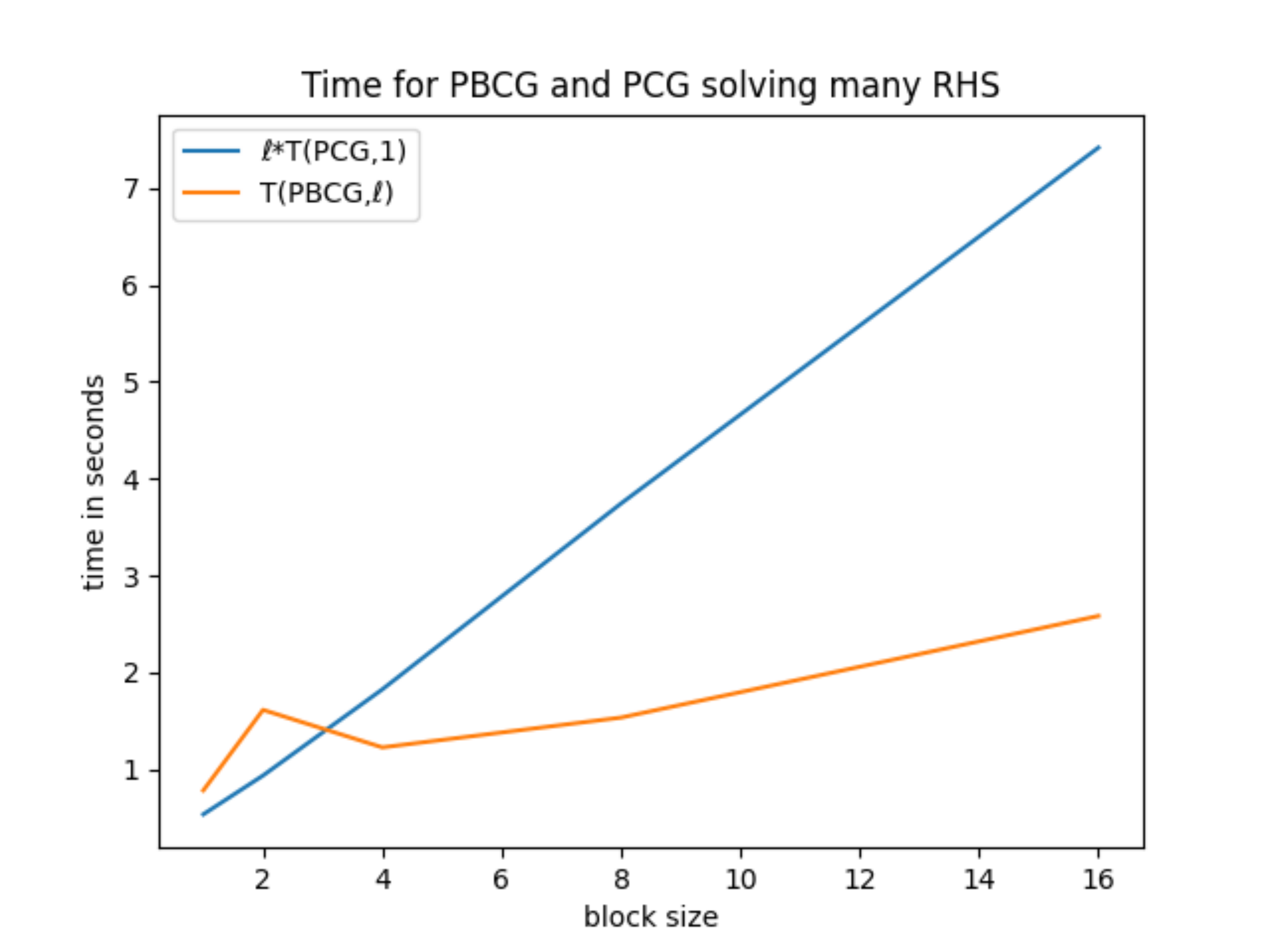


- $T_{\{PCG,\ell\}} = Iter_{\{PCG,1\}} \boxed{\cdot T_{\{MatVec,1\}} \cdot \ell}$
- $T_{\{PBCG,\ell\}} = Iter_{\{PBCG,\ell\}} \cdot \boxed{T_{\{MatMat,\ell\}}}$

$$T_{\{MatMat,\ell\}} = T_{\{A,\ell\}} + T_{\{P,\ell\}}$$

Solving ℓ linear systems using PBCG once is faster than using PCG ℓ times.

$$A_{\{n\}}, n \in \{882, 3362, \mathbf{13122}\}$$



Conclusions

- Solving ℓ linear systems using BCG or PBCG once is faster than using CG or PCG ℓ times separately. We need fewer iterations, and each iteration is cheaper for BCG when the linear system is large.
- The larger the Block size, the fewer the number of iterations.
- A well chosen preconditioner lead to fewer iterations.
- Using Pseudo Inverse can deal with singular matrices.

Accomplishment

- Fast iterative methods and concepts
- Implementation and testing of four algorithms

Future work

- Block versions of other iterative methods. e.g. GMRES.
- Use specialized routine for the sparse matrix & dense matrix product.
- Integrate solver in to PDE code and make code publicly available.

Reference

[1] Rowan Cockett. The block conjugate gradient for multiple right hand sides in a direct current resistivity inversion. 2015.

[2] Howard C Elman, David J Silvester, and Andrew J Wathen. Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics. Numerical Mathematics and Scie, 2014.

[3] Dianne P O'Leary. "The block conjugate gradient algorithm and related methods". In: Linear algebra and its applications 29 (1980), pp. 293–322.

[4] Jonathan Richard Shewchuk et al. An introduction to the conjugate gradient method without the agonizing pain. 1994.

Questions?

Thank You